CSP-J 初赛模拟卷05

数学概念

单调递增：指一个序列（数组）内的元素从左到右一直递增。单调性：我们称一直递增/一直递减的序列（数组）具有“单调性”。

一、单选题（每题 2 分，共 30 分）

1. 在 8 枚硬币中有一枚质量不合格的硬币（质量过重），你有一个无刻度的天平，最少称重几次可以找到这枚不合格的硬币？

   A. 3　　B. 2　　C. 1　　D. 4

2. 分辨率为 1024 x 900、16 位色的位图，存储图像信息所需的空间为 (　　)。

   A. 1800KB　　B. 1024 KB　　C. 4320 KB　　D. 2880 KB

3. 在一条长度为 1 的线段上随机取两个点，若进行无限次此随机操作，则以这两个点为端点的线段的平均长度是（　）。

   A. 1 / 2　　B. 1 / 3　　C. 2 / 3　　D. 3 / 5

4. 现在有 3 位男生和 3 位女生共 6 位同学站成一排，若男生甲不站两端，3 位女生中有且只有两位女生相邻，则排队方案有（　）种

   A. 360　　B. 288　　C. 216　　D. 96

5. 设 G 是有 n 个结点、m 条边 (n ≤ m) 的连通图，必须删去 G 的 (　　) 条边，才能使得 G 变成一棵树。

   A. m–n+1　　B. m-n　　C. m+n+1　　D. n–m+1

6. 已知数组 int A[8]，那么 &A[5] 的值和哪一项相等（　　）。

   A. A+160　　B. A+40　　C. A + 5　　D. A + 20

7. 线性表若采用链表存储结构，要求内存中可用存储单元地址（　）。

   A. 必须连续　　B. 部分地址必须连续　　C. 一定不连续　　D. 连续不连续均可

8. 将 7 个名额分给 4 个不同的班级，允许有的班级没有名额，有 (　　) 种不同的分配方案。

   A. 84　　B. 120　　C. 96　　D. 20

9. 算法的优劣和以下哪个要素有关？

   A. 运行算法所用的计算机　　B. 描述算法的计算机语言　　C. 算法复杂度　　D. 描述算法的代码的长度

10. 在 8 位二进制补码中，10101011 表示的数是十进制下的 (　　)。

    A. 43　　B. -85　　C. -43　　D. -84

11. 为了统计一个非负整数的二进制形式中 1 的个数，代码如下：

```
int CountBit(int x) {
    int ret = 0;
    while (x) {
        ret++;
        ________;
    }
    return ret;
}
```

则空格内要填入的语句是（　　）。

A. x >>= 1　　B. x &= x – 1　　C. x |= x >> 1　　D. x <<= 1

12. 以下哪一项不是因特网提供的服务

A. WWW　　B. E-Mail　　C. HTML　　D. FTP

13. 一个二叉树的后序遍历序列为 BEDCA，中序遍历序列为 BADEC，则先序遍历序列为：

A. ABCDE　　B. ABDCE　　C. ABCED　　D. ABDEC

14. 以下关于图的说法中，哪一项是**正确**的？

A. 在一个包含 n 个点，m 条边的无重边和自环的有向图中，某一个点的入度最多为 m。

B. 在一个包含 n 个点，m 条边的无重边和自环的有向图中，某一个点的出度最多为 n - 1。

C. 完全图中每两个点都一定互相可达，非完全图中每两个点都一定互相不可达。

D. 在一个无重边和自环的有向图中，最小的环上至少包含三个点。

15. 以下关于栈与队列的说法中，哪一项是**错误**的？

A. 栈只支持在一端进行操作，而队列支持在两端进行操作。

B. 栈是一个先进后出的数据结构，而队列是一个先进先出的数据结构。

C. 栈和队列都是线性数据结构

D. 可以用数组实现栈，但是不能实现队列。

二、阅读程序（除特殊标明外，判断题每题 1 分，选择题每题 4 分，共 40 分）

阅读程序第一题

```
01  #include <bits/stdc++.h>
02  using namespace std;
03  int main(){
04      int n, i = 2;
05      cin >> n;
06      for(; n % i != 0; i++);
07      cout << (i == n ? "Yes" : "No");
08  }
```

判断题

1. 如果将第六行末尾的分号变成一对空的大括号，程序运行结果不变。
2. 在第 7 行中如果 i 等于 n，则会输出 Yes，否则输出 No。
3. 若将第四行改为 `int n, i;`，程序运行结果一定不变。

选择题

1. 若输入为 3，则程序运行结果为

   A. NoYes　　B. NoNoYes　　C. Yes　　D. No

2. 将第 6 行改为以下哪一项，程序运行结果一定不变？

   A. `for(int i = 2; n % i != 0; i++);`

   B. `for(; sqrt(n) % i != 0; i++);`

   C. `for(; n % i != 0; i += 2);`

   D. `for(i = sqrt(n); n % i != 0; i++);`

阅读程序第二题

```
01    #include <iostream>
02    #include <string.h>
03    using namespace std;
04    char s1[100], s2[100];
05    int main(){
06        cin >> s1 >> s2;
07        int now = 0, len1 = strlen(s1), len2 = strlen(s2);
08        for(int i = 0; i < len2 && now < len1; i++)
09            if(s2[i] == s1[now])  now++;
10        if(now == len1)
11            cout << "YES" << endl;
12        else
13            cout << "NO" << endl;
14    }
```

判断题

1. 若将第 4 行改为 `string s1, s2;`，则程序运行结果一定不变。
2. 若将第 9 行改为 `if(s2[i] == s1[now++]);`，则程序运行结果一定不变。
3. 若将第 8 行中的 `&&` 改为 `||`，则程序一定会死循环。

选择题

1. 加上 `#include <cstdio>` 头文件之后，将第 6 行改为以下哪条语句，程序运行结果一定不变？

   A. `scanf("%c%c", s1, s2);`

   B. `scanf("%s%s", s1, s2);`

   C. `scanf("%c%c", &s1, &s2);`

   D. `scanf("%s%s", &s1, &s2);`

2. 当输入为以下哪个选项时，程序输出结果为 NO？

   A. `123 123`

   B. `abcdr dr`

   C. `AEW AAAEEEWWW`

   D. `abc abcdefg`

阅读程序第三题

```
01    #include <iostream>
02    using namespace std;
03    const int maxn = 100;
04    int a[maxn], ans;
05    void dfs(int l, int r){
06        if(l > r)
07            return;
08        int max_pos = l;
09        for(int i = l; i <= r; i++){
10            if(a[i] > a[max_pos])
11                max_pos = i;
12        }
13        ans += (r - l + 1) * a[max_pos];
14        dfs(l, max_pos - 1);
15        dfs(max_pos + 1, r);
16    }
17
18    int main(){
19        n = 5;
20        for(int i = 1; i <= n; i++){
21            cin >> a[i];
22        }
23        dfs(1, n);
24        cout << ans << endl;
25    }
```

注意，本题所有输入均为正整数

判断题

1. 若在 15 行之后加上一句 `return;`，程序运行结果一定不变。
2. 若将第 8 行改为 `int max_pos = r`，程序运行结果一定不变。
3. 若将第 20 行改为 `for(int i = 0; i < n; i++){`，且将第 23 行改为 `dfs(0, n-1)`，程序运行结果一定不变。

选择题

1. 若将第 6 行改为 `if(l >= r)`，则下列说法正确的是：

   A. 程序运行结果一定不变

   B. 程序可以运行出结果，且结果某些情况下会改变，某些情况下不变

   C. 程序无法运行出结果

   D. 程序可以运行出结果，且结果一定会改变

2. 若输入为 `2 1 5 2 3`，则输出为

   A. 38　　B. 29　　C. 25　　D. 13

3. 若输入为 `1 4 4 4 1`，则输出为

   A. 14　　B. 42　　C. 62　　D. 22

4. （3 分）该代码最坏情况下的时间复杂度，平均情况下（数组元素大小随机）的时间复杂度分别为?

A. $O(nlogn), O(nlogn)$　　B. $O(n^2), O(n^2)$　　C. $O(n^2), O(nlogn)$
D. $O(nlogn), O(n)$

三、程序填空（每题 3 分，共 30 分）

1.（切割）将一个单调递增的数组从某一位置切成两半，然后交换左半边和右半边。现在给出交换后的数组，多次询问，每次给出一个元素 k，问数组中是否存在该元素。

提示：先使用二分查找，找到切割点，则切割点左边的数组单调递增，右边的数组单调递增，再对半边数组进行二分查找，寻找元素 k。

```
#include <iostream>
using namespace std;
const int maxn = 1e5 + 5;
int a[maxn];
bool binary_search(int l, int r, int k){ // 用于实现二分查找，判断 [l,r] 中是否有 k 出现
    while(**1**){
        int mid = (l + r) >> 1;
        if(a[mid] == k){
            **2**;
        }else if(a[mid] > k){
            r = mid;
        }else{
            l = mid;
        }
    }
    return a[l] == k || a[r] == k;
}
int main(){
    int n, m, k;
    cin >> n;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    int L = 1, R = n, pos;
    while(L <= R){
        int mid = (L + R) / 2;
        if(**3**){
            L = mid + 1;
            pos = mid;
        }else{
            R = mid - 1;
        }
    }
    cin >> m; // 表示共有 m 次询问
    while(m--){
        cin >> k;
        if(k >= a[1]){
            if(**4**) cout << "YES" << endl;
            else  cout << "NO" << endl;
        }else{
            if(**5**) cout << "YES" << endl;
            else  cout << "NO" << endl;
        }
```

```
44        }
45    }
```

1. 1 处应填

   A. `l <= r`　　B. `l < r`　　C. `l + 1 < r`　　D. `l >= r`

2. 2 处应填

   A. `break`　　B. `return true`　　C. `l = mid + 1`　　D. `r = mid - 1`

3. 3 处应填

   A. `a[mid] > k`　　B. `mid > k`　　C. `a[mid] >= a[1]`　　D. `a[mid] > a[1]`

4. 4 处应填

   A. `binary_search(L, R, k)`

   B. `binary_search(1, pos+1, k)`

   C. `binary_search(L, pos, k)`

   D. `binary_search(1, pos, k)`

5. 5 处应填

   A. `binary_search(pos, R, k)`

   B. `binary_search(pos+1, R, k)`

   C. `binary_search(pos+1, n, k)`

   D. `binary_search(L, R, k)`

2.（井字棋）给出一个 3*3 的棋盘，双方进行下棋。先手为 player1，后手为 player2，两人轮流行动，每次行动都可以向空位下一步棋子。当某一方的棋子连成三个时（横着连成三个，竖着连成三个，或者对角线连成三个）则宣布获胜。现在给出一个井字棋的残局，如果下棋双方都很会玩，请问先手必胜，后手必胜，还是平局。（输入包含 3 行 3 列共 9 个整数，描述一个棋盘的残局，其中 0 代表棋盘中的空位，1 代表 player1 下过的棋，2 代表 player2 下过的棋）

提示：用递归来模拟下棋的过程，递归层数为奇数时，轮到 player1 下棋，否则轮到 player2 下棋。每次进入递归之前先判断当前局面是否已经有胜者出现（即分别判断 player1 和 player2 是否有棋子连成三个），否则用两重 for 循环来枚举下一个位置下在哪。注意，由于双方都很会玩，所以 player1 和 player2 都会尽量采取能够让自己获胜，不能获胜则尽量平局的策略。

```cpp
#include <iostream>
using namespace std;
int vis[4][4], cnt;
bool check(int player){ // 判断当前局面 player 是否获胜
    bool LDiagonal = true, RDiagonal = true; // 表示对角线上是否连成一条线
    for(int i = 1; i <= 3; i++){
        if(vis[i][i] != player)
            LDiagonal = false;
        if(**1**)
            RDiagonal = false;
        bool row = true, col = true; // 表示当前行和列是否连成一条线
        for(int j = 1; j <= 3; j++){
            if(vis[i][j] != player)
                row = false;
            if(vis[j][i] != player)
                col = false;
        }
```

```
18          if(row || col)
19              return true;
20      }
21      return **2**;
22  }
23  int dfs(int dep){
24      if(check(1))    return 0;
25      else if(check(2))   return 2;
26      else if(**3**)  return 1;
27      int ans = (dep % 2 == 1) ? 2 : 0;
28      for(int i = 1; i <= 3; i++)
29          for(int j = 1; j <= 3; j++){
30              if(vis[i][j])   continue;
31              vis[i][j] = (dep % 2 == 1) ? 1 : 2; // 当前 player 进行一次下棋
操作
32              if(dep % 2 == 1) // 轮到 player1 下棋
33                  **4**;
34              else
35                  ans = max(ans, dfs(dep + 1));
36              **5**;
37          }
38      return ans;
39  }
40  int main(){
41      for(int i = 1; i <= 3; i++)
42          for(int j = 1; j <= 3; j++){
43              cin >> vis[i][j];
44              if(vis[i][j] == 0)
45                  cnt++;
46          }
47      string res[] = {"player1", "draw", "player2"};
48      cout << res[dfs(1)] << endl;
49  }
```

1. 1 处应填

   A. `vis[i][i+1] != player`　　B. `vis[i][i+1] != 0`　　C. `vis[i][4-i] != player`　　D. `vis[i][i+2] != 0`

2. 2 处应填

   A. `false`　　B. `true`　　C. `LDiagonal || RDiagonal`　　D. `LDiagonal && RDiagonal`

3. 3 处应填

   A. `check(3)`　　B. `dep == cnt`　　C. `dep == cnt + 1`　　D. `dep == 3*3`

4. 4 处应填

   A. `ans = min(ans, dfs(dep + 1))`

   B. `ans = max(ans, dfs(dep + 1))`

   C. `ans |= dfs(dep + 1)`

   D. `ans += dfs(dep + 1)`

5. 5 处应填

A. `vis[i][j] = 0`

B. `cnt--`

C. `vis[i][j] = 1`

D. 什么也不用填